Signal Processing in Smart Sensor Systems 2

Milan Mišeje, Ján Šturcel

Abstract
The article refers to the previous part that is concerned in the classic methods for pattern recognition tasks in smart sensor systems. This part deals with application of the artificial intelligence means to this problem. The example given in the first part is solved using several selected types of artificial neural networks and fuzzy system. The purpose of the article is not to aim to explain the principles of these means to the readers, who have never met with them before. Only several properties related with the solving problem are mentioned. Presented solutions are summarized in term of several key factors crucial for realization in smart sensor systems.

Keywords: sensor signal processing, sensor system, artificial neural network, fuzzy system, pattern recognition

Introduction

Nowadays, the usage of smart sensor systems in many areas, mainly in automation technology, becomes as a matter of course, even as a necessary requirement. Their qualitative properties are up to standard for demanding applications. However, the abilities that could be supported by smart sensor systems are still not exhausted. They are limited only by the current state of digital technology.

Intelligence is a phenomena, which is associated usually with the human brain. In engineering, an intelligent device can be built up from artificial intelligence means, which are based upon a study of the living nature. The power of these means results from their universal properties. A required behavior can be achieved by learning them in a learning process even to learn them how to learn by yourself. From a technical and realization point of view they consist of simple small functional units (i.e. neurons), which do not require relative high computational power of the smart unit. It already appears a large variety of smart devices like neurochips, neurocomputers or fuzzychips \cite{Bouras, Suyama}, which are hardware accelerating the algorithms of artificial intelligence.

In many technical spheres the tendency is to improve qualitative properties of produced devices by implementing artificial intelligence means into them. But the reason could be else than to build up an intelligent device able to make decisions. Sensor technology is one of those areas in which these means are something new. Their usage in smart sensor systems could improve for example sensor signal processing. The idea of this article is to show such applications to pattern recognition tasks. The authors in describe their idea that sensor systems will even control production processes as agent systems do. Conceptions, mentioned above, require effective processing and evaluating of a large amount of information. For this purpose, artificial intelligence means are about to be used \cite{Allgood}.

The article follows from the previous part \cite{Miseje} and presents solution of the given example using several types of artificial intelligence means.

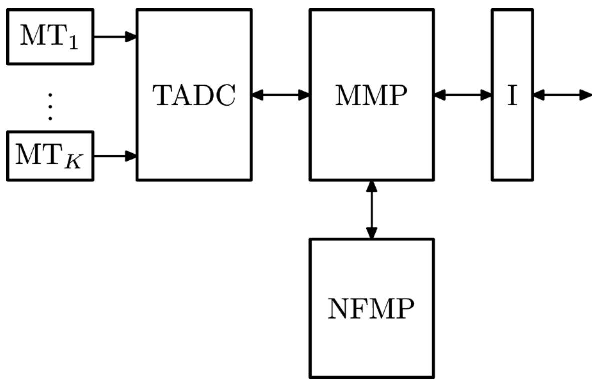
1. Artificial Intelligence Means

It exists a lot of references concerned in the principles and application examples of artificial intelligence means. The well known, most used and most propagated artificial intelligence means are *artificial neural networks*, *fuzzy systems* and *databases*. Thanks to universal properties of these means their potential is very high. Usually, they are used for pattern recognition (as a universal classificator) or for behavior modeling (as a universal approximator). Both tasks are solved in sensor technology too.

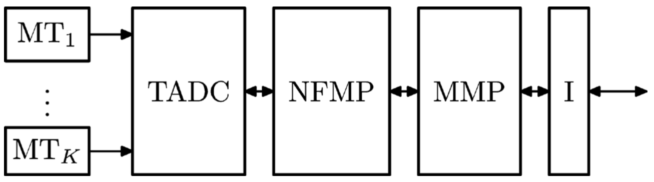
In last few years, it have become very popular especially neuro-fuzzy systems, well known as *ANFIS* (Adaptive Network-based Fuzzy Inference System). ANFIS is actually a fuzzy system, which is transformed to a neural network, when determining its parameters [6]. Then, any training method can be used to train the network. That means a fuzzy system is able to learn a required behavior and behave like a neaural network, even better.

Selection of neuro- or fuzzy-chips has been significantly increased in the last years. Their important properties like performance standard and especially miniaturization are essential for implementation into smart sensor systems. Figures Fig.1a and Fig.1b show two possible connections of a neuro-fuzzy processor in the structure of a smart sensor system. In the block diagram according to Fig.1a the neuro-fuzzy processor *NFMP* works as a coprocessor. The main microprocessor *MMP* forwards data to NFMP and NFMP returns processed data back to MMP.

According to Fig.1b, NFMP is connected directly into measuring channel. In such a connection NFMP could realize several primary information processing tasks: data reduction, filtration, linearization, dynamic error correction, disturbance correction and diagnostics. Then, the MMP would only realize communication with its environment or autocalibrating functions. The sensors S_1 to S_K makes the sensor array, *TADC* is an analog-digital converter together with a multiplexer and *I* is the communication interface of the smart sensor system.



a)



b)

Fig.1 Neuro-fuzzy processor built into the smart sensor system: a) coprocessor, b) preprocessor

Tab.1 The normalized reference patterns

	S_1	S_2	S_3
$\boldsymbol{R_1}$ (B_1)	0.1839	0.9810	0.0613
$\boldsymbol{R_2}$ (B_2)	0.8148	0.3621	0.4527
$\boldsymbol{R_3}$ (B_3)	0.0000	0.6549	0.7557
$\boldsymbol{R_4}$ (B_4)	0.8144	0.3563	0.4581

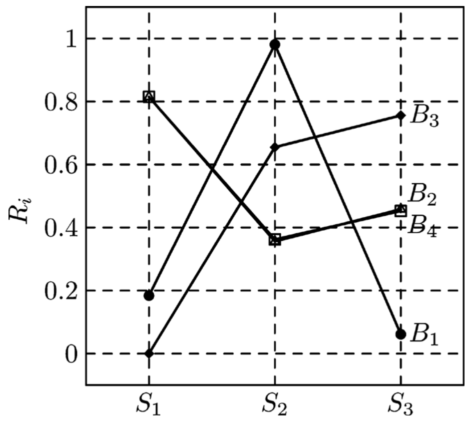


Fig.2 Graphical reprezentation of normalized reference patterns

2. Example on Identification

In the next sections, there will be presented solutions for example given in the first part [1] using several selected types of artificial intelligence means. In short, let sensor array consist of 3 sensors, which are dedicated to identify (recognize) 1 of 4 scene statuses. The evaluation will be based on the pattern recognition (PARC) principle.

Solving identification problems of PARC is available using several types of classificators. In the previous part [1] the *correlation method* as a representative of the classic methods was presented. The reference and the actual patterns were preprocessed in the same way.

Next, an application example for an actual pattern with the elements measured with precision of 2% was evaluated. Though the absolute value of sensor signals contains some kind of information, it was supressed at evaluation of correlation coefficient.

Except of bias removal, there exist *normalization* (1) to eliminate the vector length

$$\hat{\mathbf{A}} = \frac{\mathbf{A}}{\|\mathbf{A}\|} \qquad (1)$$

The vector's space orientation, which contains the needed information, remains without change and the normalized vector is a vector with length of one. Vector normalization is a native procedure needed by some neural networks like self-organised maps or networks with radial basis functions, which are discussed later in the article.

The normalized reference patterns (Tab.1, Fig.2) were created from the beginning of ranges (0%) of physical quantities. It is obvious that they might be different from those created from the end of the ranges (100%). Therefore, when dealing with pattern recognition of signals, which values are changing in some interval, normalization is not a good choice for length elimination. Furthermore, the computational demand on vector normalizing is greater than on bias removal. An actual pattern, of course, should be also preprocessed the same way as the reference patterns.

The similarity of two patterns **X** and **Y** can be evaluated by known Euclidean formula that gives the distance *d* between these patterns

$$d^2 = \sum_{i=1}^{N} (x_i - y_i)^2 \qquad (2)$$

Euclidean distances between the normalized reference patterns are in table 2.

Tab.2 Euclidean distances between normalized patterns

	$\boldsymbol{R_1}$	$\boldsymbol{R_2}$	$\boldsymbol{R_3}$	$\boldsymbol{R_4}$
$\boldsymbol{R_1}$	0	0,9665	0,7889	0,9722
$\boldsymbol{R_2}$	0,9665	0	0,9173	**0,0080**
$\boldsymbol{R_3}$	0,7889	0,9173	0	0,9170
$\boldsymbol{R_4}$	0,9722	**0,0080**	0,9170	0

From table Tab.2 it results that identification of statuses B_2 and B_4 can be problematic because patterns $\mathbf{R_2}$ and $\mathbf{R_4}$ are almost identical. Euclidean distance between them is relatively small. To avoid incorrect results, considerably more precise sensors would have to be used. The absolute error Δ_A of each sensor is determined from minimal Euclidean distance d_{min} between reference patterns according to

$$\Delta_A = \frac{1}{2}\frac{d_{\min}}{\sqrt{N}} \qquad (3)$$

where *N* is number of sensors. It holds only for non-normalized vectors.

In the following sections, there will be described solutions of the example using MLP, RBF and SOM neural networks and using neuro-fuzzy system. As there exist patterns of which normalized vectors are allmost identical, it will be searching for such approaches, which do not need normalization even any other preprocessing of measured signals from the sensor array. At the beginning of each section, there will be in short described the basic principles of the selected means. For deeper study of their properties and possibilities it is available [5].

2.1 Solution using MLP neural network

Neural networks have been studied for several decades since Rosenblatt first time applied single-layer perceptron to learning the pattern recognition in 1950. Perceptron represents one of the first attempts to build intelligent and self-learning systems using simple components. Unfortunately, single-layer perceptron was not successfully applied to modeling of some logical functions. Well known is the XOR problem that has decreased the enthusiasm about perceptrons for some time. Individual subjects have to be *linearly separable*. Otherwise, there is need to expand the number of perceptron's inputs (multiplication or power of the existing), or expand the number of perceptron's layers. A larger scale of use have reached MLP (Multi-Layer Perceptron) neural network. One can say, this is the most known and most used neural network.

For the identification task will be applied three-layer perceptron neural network with input, hidden and output layer. Every neuron (\figurename~\ref{fig:mlpnn}) in the layers realizes the function

$$y = \psi\left(\sum_i w_i \cdot x_i + \theta\right) \tag{4}$$

where *ψ(·)* is activation function of neuron, w_i are connection weights, x_i are inputs, *θ* is threshold. The network is feed-forward with supervisor training. There exist several adaptation methods to network training (determining network parametrs) [5], for example, gradient descent method. First, it is starting with a set of random connection weight. Then, an input vector **x** from training data set is selected. Finally, if perceptron gives an incorrect response *y*, all connection weights w_i are modified according to

$$\Delta w_i = \eta y_i x_i \tag{5}$$

where *η* is learning rate coefficient, y_i is target output and x_i is input vector's element.

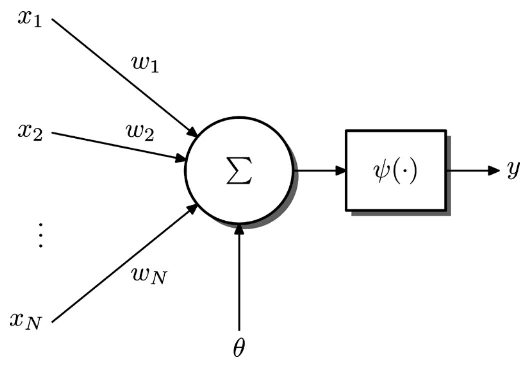


Fig.3 Structure of perceptron

From equation (4) results that evaluation of the neuron activation function *ψ(·)* could be difficult computing operation. For identification purposes two states: "IS", "IS NOT" are sufficient. Thus, the *unit step* activation function is appropriate, for which it holds

$$\psi(x) = \begin{cases} 1 & \text{ak } x > 0 \\ 0 & \text{inak} \end{cases} \tag{6}$$

This is already a simple operation, except which, only the average sum needs to be evaluated in the algorithm. So, all together are very simple without large computational demands on realization. The algorithm of the network can be emulated in a classic monolithic microcomputer.

The solution of the given example is practically not complicated. Out of example, some parameters of neural network like number of neurons in the hidden layer and number of inputs in each of these neurons can be determined. Concrete, the number of neurons is 4 and thus is equal to the number of identifiable states. The number of inputs to the neurons is 3 and is equal to the number of sensors in the sensor array. The measured data from the sensor array do not need be preprocessed before passed to neural network. This makes it simpler to implement algorithm of classificator. Each neuron will signalize the identified status by 1 and by 0 the others. The output from the hidden layer is a vector with 4 elements. The index of the element with value 1 means identified status. Another layers are: input layer - distribution of input signals to the hidden layer, output layer - evaluation of signals out of neurons in the hidden layer. For example, only one neuron in the output layer with *proportional* activation function and connection weights with values 1,2,3 and 4 respectively generates directly the index of an identified status.

The training data set should contain data from whole output range of sensors in order to cover every situation. Because all transfer characteristics are monotonic, it will be enough to regard only start and end point of ranges (input at 0 a 100% ±tolerance). It means that the training data set will contain 8 input and 8 output vectors. For example, the status B_1 gives the pair of input and output vectors as follows:

X[%]	input	output
0	$(1{,}5\ 8{,}0\ 0{,}5)^T$	$(1\ 0\ 0\ 0)^T$
100	$(3{,}0\ 9{,}5\ 2{,}0)^T$	$(1\ 0\ 0\ 0)^T$

If signals passed to the input of the network are within the range listed above, then the first element of the output vector (output of the first neuron) will be set to 1 that indicates status B_1.

Such a solution is simple, however it needs that all classes are linearly separable already by single-layer perceptron (in the hidden layer). The more complicated problems are able to be solved with more layers and training of network can be accomplished using the familiar *backpropagation method*.

2.2 Solution using RBF neural network

The structure of RBF(Radial Basis Function) neural network that will be used for PARC is depicted in figure Fig.4. The network has *K* inputs x_k representing the elements of a signal vector. The number of neurons *N* is equal to the number of reference patterns. Each of the neurons stores one reference pattern in its centre c_n. Neurons perform radial basis function *R(·)* according to equation (7) over neuron's input values and parameters. Outputs from neurons are rounded by rounding function and multiplied by weight coefficients w_n. Finaly all the values are added together in the summator Σ. The rounding functions can be replaced by perceptrons (see figure Fig.3) with hard limit transfer functions, thresholds set to 0.5 and input weight coefficients equal to the 1.

Radial basis function *R(·)* is defined as follows

$$R(x) = \exp\left(-\gamma \cdot \|\mathbf{c} - \mathbf{x}\|^2\right) \tag{7}$$

and the overall network's function is

$$y = \sum_{i=1}^{N}\left[w_i \cdot \exp\left(-\gamma \cdot \|\mathbf{c_i} - \mathbf{x}\|^2\right)\right] \tag{8}$$

where w_i are weight coefficients, $\mathbf{c}_i$ are centers of neurons, **x** is input vector and *γ* is spread or kernel.

Solving the following linear equation system usually does the learning of RBF neural network.

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} e^{\left(-\gamma\cdot\|\mathbf{c}_1-\mathbf{c}_1\|^2\right)} & \cdots & e^{\left(-\gamma\cdot\|\mathbf{c}_1-\mathbf{c}_N\|^2\right)} \\ \vdots & \ddots & \vdots \\ e^{\left(-\gamma\cdot\|\mathbf{c}_N-\mathbf{c}_1\|^2\right)} & \cdots & e^{\left(-\gamma\cdot\|\mathbf{c}_N-\mathbf{c}_N\|^2\right)} \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} \qquad (9)$$

Elements in the diagonal of the matrix of radial basis functions are 1's. The size of the matrix is equal to the number of the reference patterns. Solution of this linear equation system gives the vector of the weight coefficients **w**. Spread γ and output vector **y**, passed to the linear system, are to be chosen. Disadvantage of this learning method is that all parameters need to be recomputed by (9), when a new reference pattern will have been added.

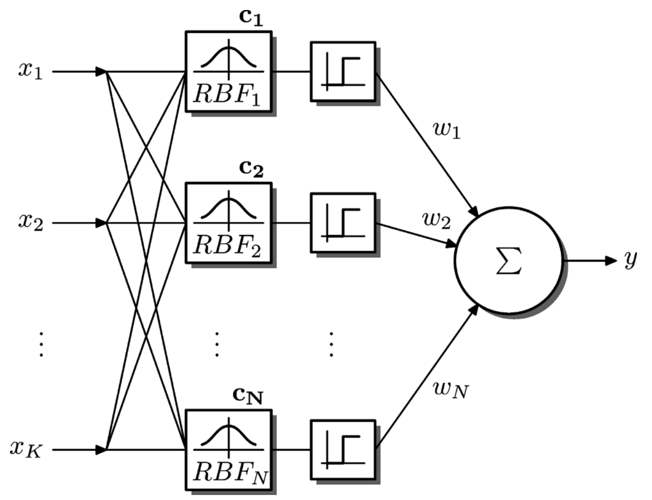


Fig.4 Structure of RBF neural network

The important role plays center selection. If a center were a vector with the elements equal to the middle of the output ranges of the particular transfer characteristics, Euclidean distance between the centre and both end points of these characteristics would be equal. In figure Fig.5, there are graphed Euclidean distances between the reference pattern for status B_2 and the patterns for all states in their whole definition scope. This way, the distances of the other reference patterns can be examined. One can find out that an appropriate spread γ makes it possible to separate the patterns each other in the whole range.

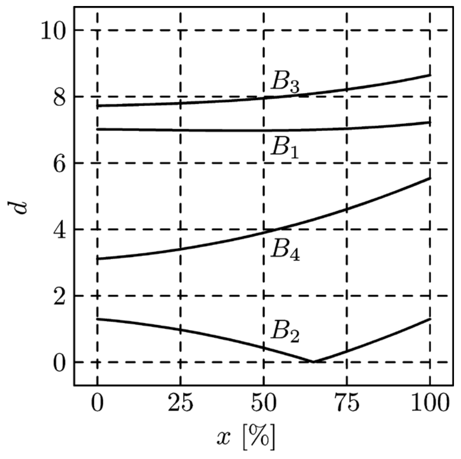


Fig.5 Euclidean distances from the reference pattern R$_2$

The learning process by (9) can be skipped only by choosing **w** and spread γ. Coeficients of **w** correspond with the outputs **y**. The greater is γ, the sharper are radial basis functions around center $\mathbf{c_i}$ and network will be more sensitive to noisy input signals. It is possible to bypass any sensor signal preprocessing when utilizing RBF network as well as MLP networks.

2.3 Solution using SOM neural network

SOM stands for self-organizing map and this category of neural network will be represented by Kohonen neural network (Fig.6). SOM networks are specific in training, which does not need a supervisor. The number of inputs to the network is equal to the dimension of the input signal vector, which make up the patterns. Patterns are encoded into the connection weights of the inputs. SOM network is usually a planar grid, in whom the neurons are the outputs at the same time. Thus the number of outputs is equal to the number of neurons. The outputs of neurons in neighbour are connected each other. The one and only difficult operation that neurons perform is computing the distance between a passed pattern and the patterns encoded in the connection weights, according to the Euclidean relationship (2).

At the beginning of the learning process all connection weights are set to the small random values. So, each neuron will have got its own original weight vector. During the learning process, neighbourhood of the winning neuron and learning rate coefficient are slowly decreasing. In the evaluation process the network is searching for a neuron which is closest to the input data. Then the group or class, in which the neuron belongs to, is determined. Any unknown patterns passed to the network will be categorised to the nearest clusters. This property is known as *generalization*.

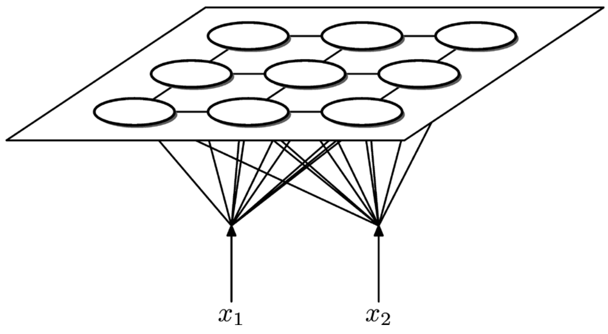


Fig.6 Structure of SOM neural network

The neural network uses Euclidean metrics. For pattern recognition the vector space orientation is more important than the absolute value of the vector elements. Therefore the vector length is usually eliminated. Vector normalizing was described in the section "Example on Identification". Realization of vector normalizing in smart sensor system is relatively difficult computing operation. Even when the root is not needed be calculated in case of searching for the closest pattern.

A possible solution to the given example is as follows; From figure Fig.5 and by analysis of all reference patterns the same way it results that there is enough space between every reference pattern and any pattern for other status. This allows skipping any signal preprocessing like vector normalization. So, the network will be made from 4 neurons aligned in the 1x4 array. Thus, each neuron is in neighbourhood with 2 other neurons except the outers. Training data set contains 4 vectors $\mathbf{v_i}$ each of that have 3 elements (the number of inputs to the network is equal to the number of sensors):

	$\mathbf{v_1}$	$\mathbf{v_2}$	$\mathbf{v_3}$	$\mathbf{v_4}$
S_1	2,25	5,25	0,75	8,75
S_2	8,75	2,75	7,25	4,25
S_3	1,25	3,25	8,25	5,25

Elements are equal to the middle of the output ranges of particular transfer characteristics, so Euclidean distances to the end points are identical.

The network output is additionally defined because after every initialization (setting of connection weights to the random values) and network training the neurons can be active for different classes.

2.4 Solution using fuzzy system

Block scheme of a fuzzy microprocessor is depicted in figure Fig.7 that clearly explains the functionality of fuzzy system. Input variable *x* has crisp value to which fuzzy sets are applied in the block *fuzzification*. Next it follows the *inference mechanism* block that evaluates all the fuzzy rules stored in the *fuzzy rule base* block. The fuzzy rule base is fully programmable through the programming interface. The *operating memory* block is used at every operation. The *defuzzifikation* block produces crisp outputs *y* from the fuzzy values that are the outputs of the inference mechanism block. Signal conversions are supported by the analog-digital *ADC* and digital-analog *DAC*. Appropriate fuzzy models for solved problem are Mamdani- or Sugeno-type fuzzy system. They differ in the consequents of the fuzzy rules, where the Mamdani-type uses singletons or fuzzy sets and Sugeno-type uses constants or polynomial functions.

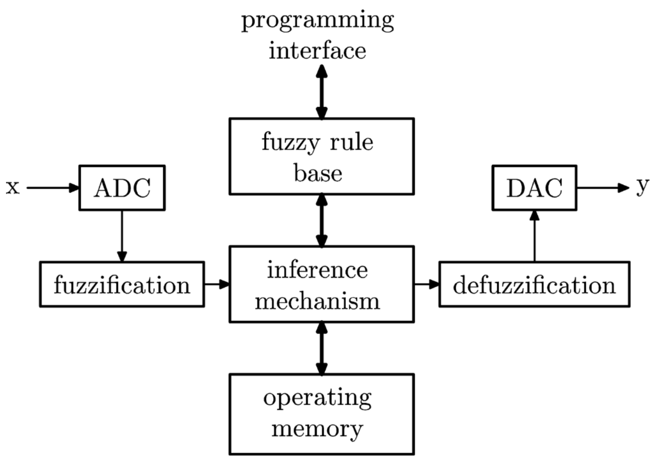


Fig.7 Block scheme of fuzzy microprocessor

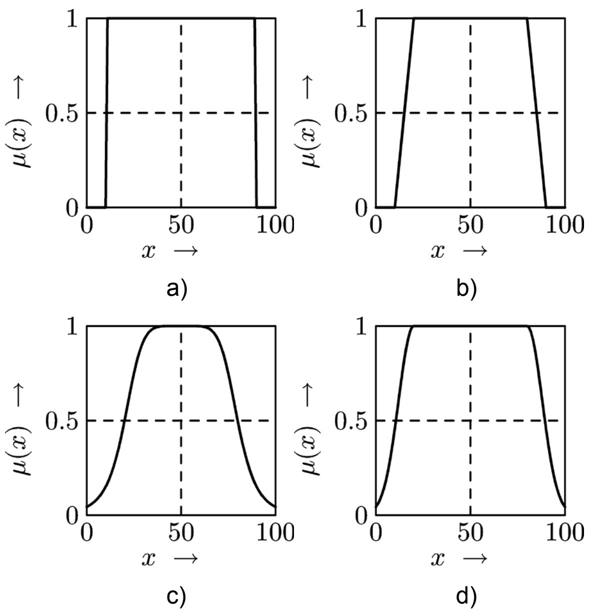


Fig.8 Membership functions: a) rectangular, b) trapezoidal, c) bell, d) two-sided gaussian curves

Since the sensor output signals are changing within particular intervals according to transfer characteristics, it is possible that these intervals determine parameters of the fuzzy sets. In order to get a simple solution of given example, it will be considered membership functions μij(x) with rectangular curves (Fig.8a), for which it holds:

$$\mu_{ij}(x) = \begin{cases} 1 & \text{if } x \in \langle a_{ij}; b_{ij} \rangle \\ 0 & \text{else} \end{cases}$$

where a_{ij} and b_{ij} are the boundaries of the *i*-th sensor output signal interval for the *j*-th scene status. Similar membership function curves are trapezoidal (Fig.8b), bell (Fig.8c) or two-sided gaussian (Fig.8d) curves.In the evaluation process, it is usually used the firing strength of a rule instead of direct usage of the output value [5]. The higher firing strength of a rule is, the higher is validation of the consequent of a fuzzy rule. The solution presented next utilizes direct output of the fuzzy system.

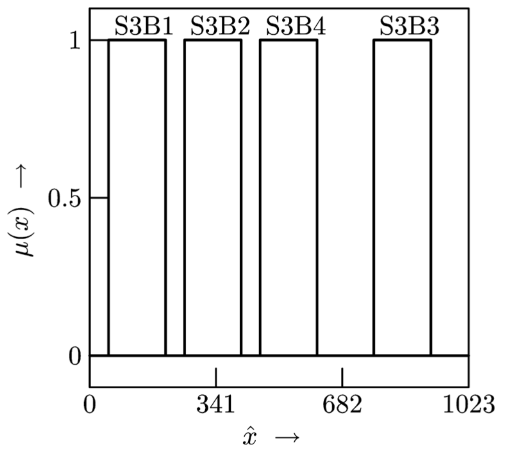


Fig.9 Fuzzy sets for signal from sensor S_3

Though fuzzy sets can overlap one another, at some circumstances overlapping of fuzzy sets might cause classes count reduction or ambiguous results. In figure Fig.9 are shown the fuzzy sets for sensor S_3. The symbolic name *S3B1*, for example, stands for fuzzy set that covers signals from sensor S_3 when status B_1 appears. It is obvious that fuzzy sets are not overlapping one another. That means the output signal of sensor S_3 would be enough for status identification. However, this is only an exception because of simplicity of the given example. Parameters of fuzzy sets are listed in Tab.3. In the first column are listed symbolic names of fuzzy sets. The second column contains intervals of output values of measuring transmitter with sensor S_3. And the intervals of $\hat{x}$ obtained from 10-bits ADC are listed in the last one. Variables *a* and *b* represents the fuzzy sets parameters.

Tab.3 Parameters of fuzzy sets for sensor S_3

	U[V]	[a;b]
S3B1	<0,5;2,0>	[51;205]
S3B2	<2,5;4,0>	[256;409]
S3B3	<7,5;9,0>	[767;921]
S3B4	<4,5;6,0>	[460;614]

The number of inputs into fuzzy system is equal to the number of elements in a reference pattern. When adding a new pattern, one needs just to add several new fuzzy sets and fuzzy rules without modifying any old one. That is, neither the training of the fuzzy system, nor pattern vector normalization is needed in this case. While signals from the sensor system are incoming directly into the fuzzy system, the scheme shown in figure Fig.1b can be utilized.

Fuzzy rules of the fuzzy system are following:
if S_1 **is** S1B1 **and** S_2 **is** S2B1 **and** S_3 **is** S3B1 **then** y **is** B_1
if S_1 **is** S1B2 **and** S_2 **is** S2B2 **and** S_3 **is** S3B2 **then** y **is** B_2
if S_1 **is** S1B3 **and** S_2 **is** S2B3 **and** S_3 **is** S3B3 **then** y **is** B_3
if S_1 **is** S1B4 **and** S_2 **is** S2B4 **and** S_3 **is** S3B4 **then** y **is** B_4

where S_i are the output signals from the sensor system, S_iB_j is the fuzzy set in the antecedents of the rules for the *i*-th sensor and the *j*-th scene status and B_j is the fuzzy set in the consequent of the rules. The strength of a rule determines the output value *y*. The number of the rules is equal to 4, which is equal the number of the reference patterns.

By adding the following fuzzy rule to the inference mechanism:
if S_1 **is not** S1B1**and** S_1 **is not** S1B2 **and** S_1 **is not** S1B3 **and** S_1 **is not** S1B4 **then** y **is** D_1

one obtains a simple diagnostics that can detect failure of measuring channel with sensor S_1. Malfunction of measuring channel with sensor S_1 will be signalled by output *y* equal to D_1.

Conclusion

In this article were presented several solutions for example given in the previous part using artificial intelligence means. The obtained results show that they offer a perspective alternative to classic methods for pattern recognition tasks. Their essential properties for use in smart sensor systems are summarized and rated in the following table:

	MLP	RBF	SOM	fuzzy
operation count	+ + +	- -	-	+ +
computational demand	+ + +	- - -	- - -	+

The solutions have utilised MLP, RBF and SOM neural networks and fuzzy system, each of which is something specific. MLP neural network has a simple structure, however the classes must be linearly separable. RBF network better perform cluster analysis and training can be done in one step. SOM network creates groups or classes by yourself without supervisor training. Fuzzy system works with linguistic variables and offers simple diagnostics of measuring channels.

Absolute value of measured signals has some kind of useful information for identification purposes. It is not effective to avoid it. All presented approaches do not require any preprocessing of measured signals (normalization, bias removal) from sensor array, whereby considerable reduction of required operations have been achieved. Classificator based on MLP neural network or fuzzy system are easily applicable to emulation in classic monolithic microcomputer in smart sensor system.

When designing a neural design, the final structure, training data set and training method are very important to achieve satisfied results. It rules that network should be easy implementable, which is a important feature for emulation of the network algorithm in a classical microcomputer. Thanks to hardware devices accelerating functionality of artificial intelligence means, the intelligent unit of smart sensor system is getting less busy. Implementing such devices into smart sensor systems has the future perspective. In comparison with the classic methods, the quality of sensor signal processing will be at a higher level. Pattern recognition is an necessary step for implementation of artificial intelligence into smart sensor systems.

References

[1] Mišeje,M., Šturcel,J.: Signal Processing in Smart Sensor Systems (part 1). In: Journal of Electrical Engineering, 2003, vol. 54, no. 6–7, pp.154–159.

[2] Bouras,S., Kotronakis,M., Suyama,K.: Mixed Analog-Digital Fuzzy Logic Controller with Continuous-Amplitude Fuzzy Inference and Defuzzification. In: IEEE Transactions on Fuzzy Systems, 1998, vol. 6, no. 2, pp. 205–215.

[3] Bouras,S., Suyama,K., Tsividis,Y.: Integrated Fuzzy Logic Controller with Continuous Processing. In: Fifth IEEE International Conference on Fuzzy Systems, New Orleans, Louisiana, 1996, vol. 3, pp. 1951–1957.

[4] Allgood,G.O., Manges,W.W.: Sensor Agents – When Engineering Emulates Human Behavior. In: Sensors. [online]. 24/2 2002. [cit: 2002-08-13]. Dostupné na internete: <http://www.sensorsmag.com/articles/0801/28/main.shtml>.

[5] Jang,J.S.R., Sun,Ch.T., Mizutani,E.: Neuro-Fuzzy and Soft computing: a computational approach to learning and machine intelligence. New Jersey: Prientice Hall, Inc., 1997, ISBN 0-13-261066-3

[6] Jang,J.S.R., Sun,C.T.: Neurofuzzy modeling and control. In: IEEE Transaction on Fuzzy System, 1995, vol. 3, no. 3, pp. 378–406.

Doc. Ing. Ján Šturcel, PhD.

Slovak University of Technology
Faculty of Electrical Engineering and Information Technology
Department of Automation and Control
Ilkovičova 3
812 19 Bratislava
Tel.: (+4212 6029 1678)
jan.sturcel@stuba.sk